TRANSIMS Software Architecture for IOC-1

K. P. Berkbigler
Computer Research and Applications Group
Los Alamos National Laboratory
and
B. W. Bush
Energy and Environmental Analysis Group
Los Alamos National Laboratory
and
J. F. Davis
Software Design and Development Group
Los Alamos National Laboratory

24 April 1997

Abstract

This document describes the TRansportation ANalysis SIMulation System (TRANSIMS) software architecture and high-level design for the first Interim Operational Capability (IOC-1). Our primary goal in establishing the TRANSIMS software architecture is to lay down a framework for IOC-1. We want to make sure that the various components of TRANSIMS are effectively integrated, both for IOC-1 and beyond, so that TRANSIMS remains flexible, expandable, portable, and maintainable throughout its lifetime. In addition to outlining the high-level design of the TRANSIMS software, we also set forth the software development environment and software engineering practices used for TRANSIMS.

I. Introduction

A. Overview

This document describes the TRANSIMS software architecture and high-level design for the first Interim Operational Capability (IOC-1). It is important to note that some of the IOC-1 implementation may differ from this design and some aspects of the design may not be implemented until a future IOC—see the IOC-1 documentation for the individual TRANSIMS components for details on the implementations.

Our primary goal in establishing the TRANSIMS software architecture is to lay down a framework for IOC-1. We want to make sure that the various components of TRANSIMS are effectively integrated, both for IOC-1 and beyond, so that TRANSIMS remains flexible, expandable, portable, and maintainable throughout its lifetime. We start by breaking down each of the major IOC-1 software *systems* (Simplified HCAD, Interim Planner, Low-Fidelity Microsimulator, Input Editor, and Output Visualizer) into *subsystems*. We then define the functionality of each subsystem, but we will not concern ourselves with the design of the subsystems themselves in this document.

In addition to outlining the high-level design of the TRANSIMS software, we also identify the commercial software that forms the software development environment (i.e., compiler, debugger, configuration management tool, case tool, etc.), and the commercial software incorporated into the run-time TRANSIMS applications (i.e., C++ libraries, databases, geographic information systems, statistics packages, etc.).

Finally, we discuss recommended software engineering practices and standards for coding, documentation, methodology, configuration management, and testing.

B. The Importance of Architecture

Architectural considerations become more important the larger a software development project becomes: [Mc 93]

> On a small project, construction is the most prominent activity by far, taking as much as 80 percent of the total development time. On a medium-size project, construction is still the dominant activity but its share of effort falls to about 50 percent. On very large projects, architecture, integration, and system testing each take up about as much time as construction.

Figure 1 illustrates this point.

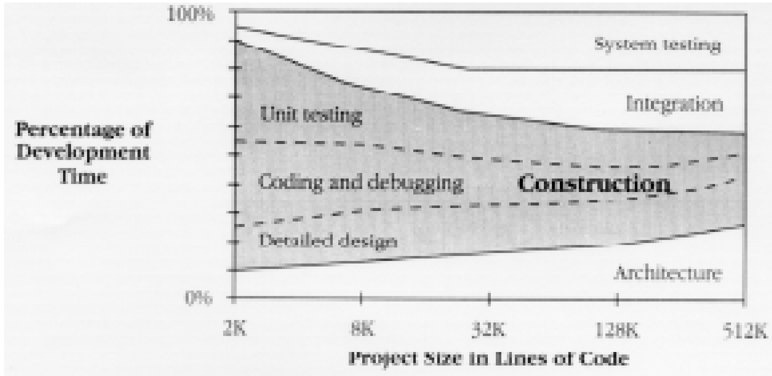

Figure 1. Activity proportions and project size. [Mc 93]

It also pays to do things right the first time: Empirical data shows that a change in the early stages of a project, in requirements or architecture, costs 50 to 200 times less than the same change later, in construction or maintenance [Mc 93]. Researchers at IBM found that purging an error by the beginning of design, code, or unit test allows rework to be done 10 to 100 times less expensively than when it's done in the last part of the process [Mc 93]. Figure 2 illustrates this point.

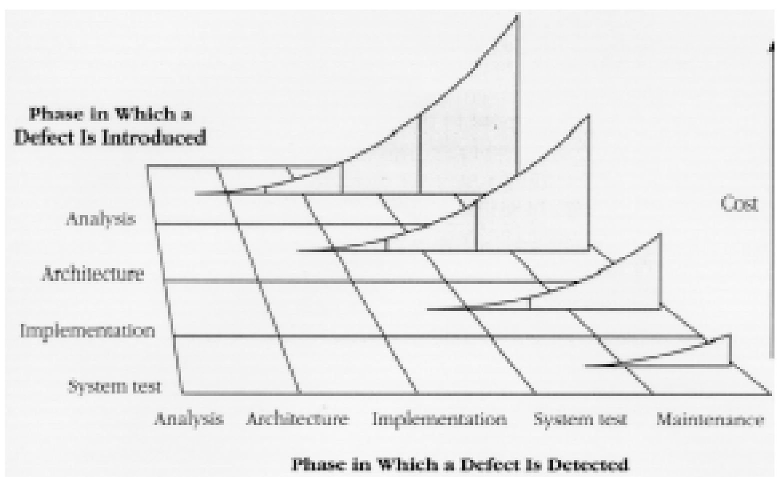

Figure 2. Cost of fixing defects. [Mc 93]

C. Architectural Principles in TRANSIMS

We have adopted four major principles of modern software engineering for the architecture of TRANSIMS: layering, modularity, iteration, and object-orientation.

Layering separates the software components into a hierarchy with the application at the top, the domain in the middle, and the technology at the bottom. Each layer uses the layers below it, but not vice versa. Layering encourages the reuse of software components in different parts of the application. Layering provides an integrated framework for the software development.

Each software component/module has responsibilities and provides services. The actual implementation of the module is separate from its public interface. *Modularity* reduces the coupling between software components that can make maintenance, reuse, portability, and extension difficult.

The iterative development process reduces risk. Each *iteration* refines the system and results in an executable release. Figure 3 illustrates the TRANSIMS iterative development process.

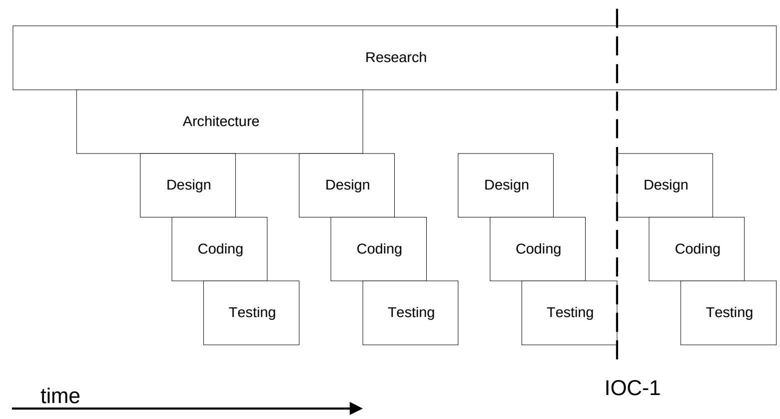

Figure 3. The TRANSIMS iterative development process

Key features of *object-orientation* [CS 94] include *abstraction/encapsulation*, which allows building models that map to the real world, *inheritance*, which enables code reuse, and *polymorphism*, which reduces software maintenance and increases extensibility. Critical issues addressed by object technology are *scheduling* (meeting delivery dates), *complexity* (modeling complex applications), *size* (managing interdependencies in large systems), and *compatibility* (making different chunks of code interoperate). The use of object technology results in reuse of code, reduced code size, increased productivity, and lower defect rate.

II. Design

The layered architecture for TRANSIMS contains the following layers:

- *Application:* The Analyst Toolbox provides a centralized interface for TRANSIMS.
- *System:* The TRANSIMS systems centralize access to the major functional components of TRANSIMS.
- *Subsystem (high-level):* The high-level subsystems each provide services to one or more TRANSIMS systems. This enhances the reusability and flexibility of the software.
- *Subsystem (low-level):* The low-level subsystems provide basic services (mostly, data and operations on data) to the high-level subsystems and to the systems. They

provide a common representation of objects such as vehicles, travelers, the road network, etc. They never directly interact with the user.

- *Subsystem (utility):* The utility subsystems provide basic domain-independent services to the higher-level components of TRANSIMS. They are used to isolate the domain subsystems from dependence on the operating system, file system, etc.

Figure 4 depicts this layered architecture; for diagrammatic simplicity, connections between the third and fourth layers are not shown. Different systems use different combinations of subsystems to implement their functionality (see Table 1).

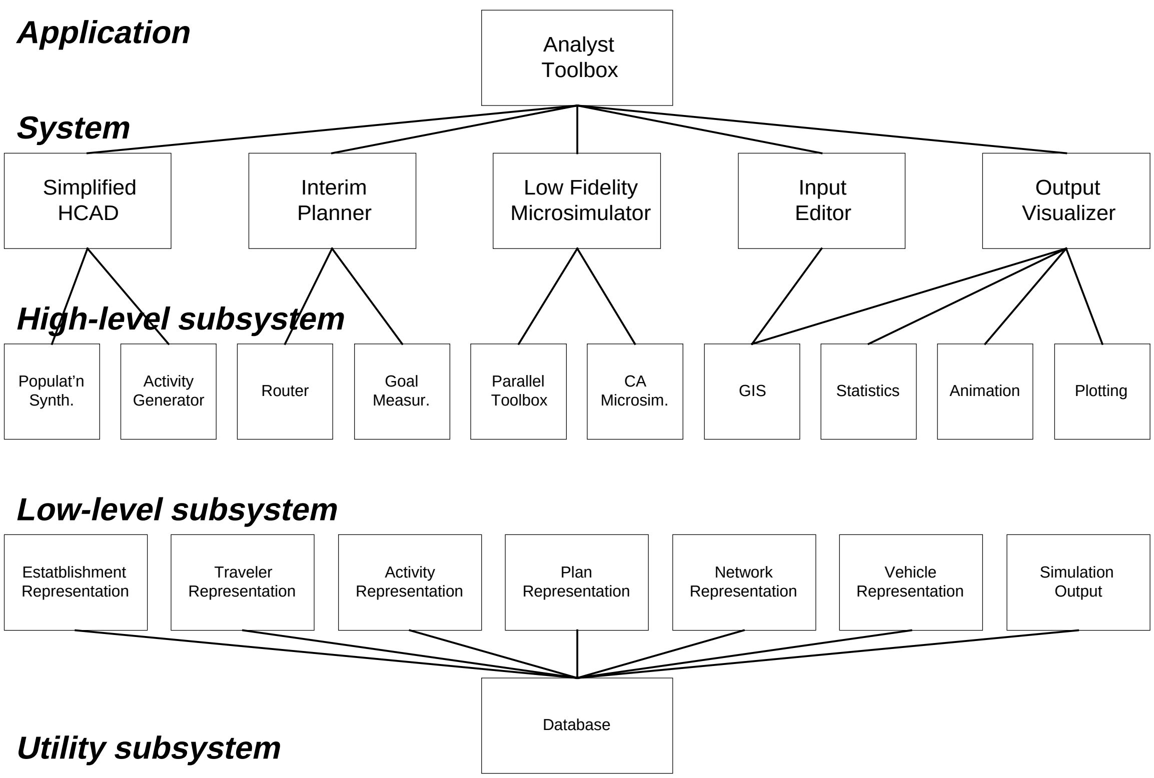

Figure 4. TRANSIMS software systems and subsystems.

Table 1. Major interdependencies between TRANSIMS systems and subsystems.

Subsystem	*Input Editor*	*Simplified HCAD*	*Simplified Planner*	*Low-Fidelity Microsimulator*	*Output Visualizer*
Network Representation	√		√	√	√
Establishment Representation	√	√	√		√
Traveler Representation	√	√	√	√	√
Vehicle Representation	√		√	√	√
Plan Representation	√	√	√	√	√
Population Synthesizer		√			
Activity Generator		√			
Origin-Destination (OD) Matrix/Route Disaggregator			√		
Router			√		
Parallel Toolbox				√	
Cellular Automaton (CA) Microsimulation				√	
Simulation Output				√	√
Geographic Information System GIS)	√				√
Statistical Analysis					√
Animation					√
Plotting					√
Database	√	√	√	√	√

A. Application Layer

The Analyst Toolbox provides a centralized and uniform interface integrating the major TRANSIMS systems. It forms the primary user interface for an analyst.

B. System Layer

The TRANSIMS systems centralize access to the major functional components of TRANSIMS. For IOC-1, there are the following systems:

- Input Editor
- Simplified Household and Commercial Disaggregator (HCAD)
- Interim Planner
- Low-Fidelity Microsimulator
- Output Visualizer

Additional systems may be added for future IOCs or from sources external to LANL. For example:

- Air Quality Estimator
- High-Fidelity Microsimulator
- Land Use Estimator

1. Input Editor

The input editor provides a means for editing the database and setting up scenarios for simulation. All scenario data will be editable. It will have functions for:

- importing new road networks
- altering existing networks
- merging networks
- extracting networks
- editing traveler characteristics
- editing establishment characteristics
- editing vehicle characteristics

The editor will be integrated into the rest of the geographic information system (GIS) software, supporting visual/graphical editing of geographic objects, table editing of non-geographic objects, and editing via ad-hoc queries (using the native query languages for ArcView, Arc/Info, and Oracle). Version control will also be supported.

Figure 5 shows how the input editor might use other TRANSIMS components.

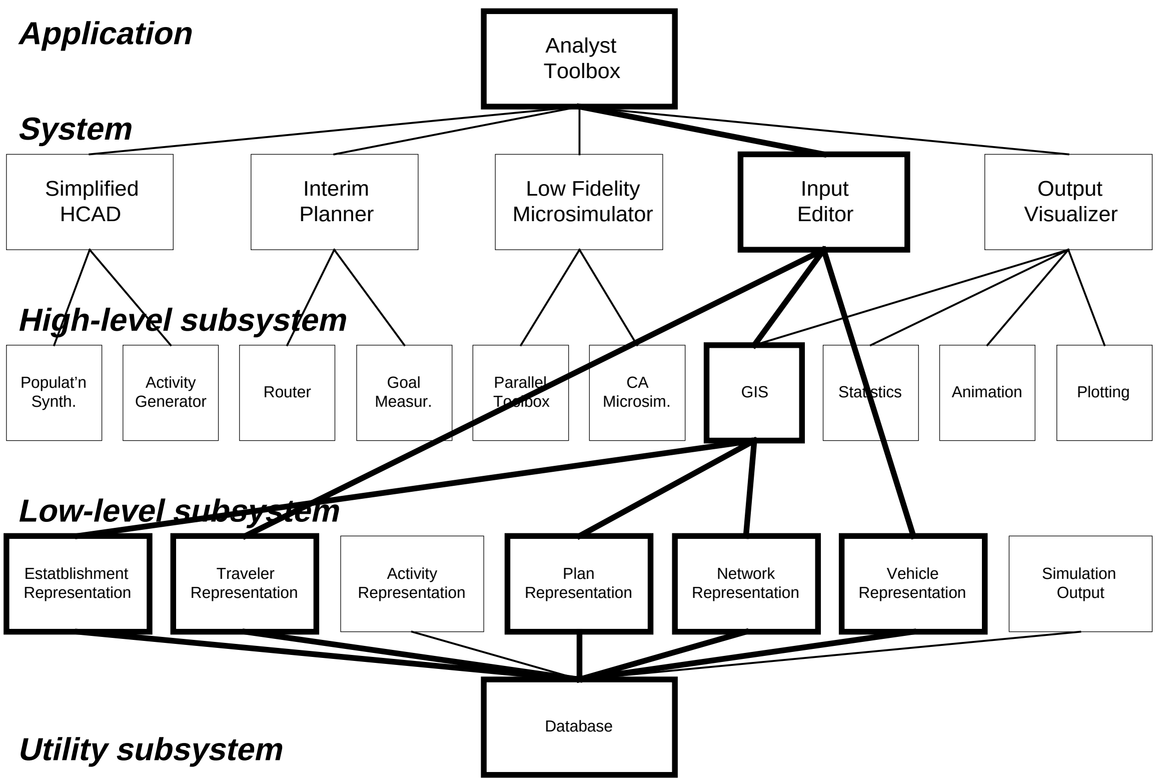

Figure 5. Software components used by the input editor.

2. Simplified Household and Commercial Activity Disaggregator (HCAD)

The simplified HCAD system provides the ability to construct travel activities for study areas. This system will have the capability to select and import any of the available synthetic populations for a study region and to generate travel activity from the chosen synthetic population. The focus in IOC-1 will be on households, as opposed to commercial activity.

3. Simplified Planner

The simplified planner will provide the microsimulator with household or individual travel demand in the form of trip plans. For IOC-1, the major effort will be to develop the data structure that integrates the flow of data from the synthetic population and its activity demand, through the planner's trip planning process, and on to the microsimulation. The planner's process, or mode/route/activity assignment algorithms, will be simplified to accommodate available data and microsimulation development time constraints.

There are three types of input data: the transportation network, household/individual activity demand, and individual travel behavior. The network required by the planner is a subset of the total network representation. Activity demand and travel behavior are derived from national and local demographic and socioeconomic characteristics, and local land use. For IOC-1, the simplified HCAD will provide this data. While the household is the basic unit of activity input, the emphasis in IOC-1 will be to identify mandatory activities at the individual level. Discretionary activities at the household or individual level may be included. Individual travel behavior input will consist primarily of a limited set of goals and preference distributions.

The trip planning process will consist of activity, mode, and route assignment algorithms. For IOC-1, no activity assignments will be made; the mandatory activities will be provided by the simplified HCAD. No modal assignments will be made because of the focus on the automobile. Carpooling will not be addressed; pedestrian movement may be addressed but only with respect to movement to and from the automobile.

Output data are the individual trip plans required by the microsimulation. This data includes trips and trip chains consisting of origins, activity destinations, routes, and times associated with activity performance and route movement.

4. Low Fidelity Microsimulator

The low-fidelity microsimulator is a regional-scale, low-fidelity traffic microsimulation based on cellular automata and implemented on a distributed computer network. It will be the primary technical focus of IOC-1. A preliminary design for this type of microsimulation was proposed by Marcus Rickert [Ri 94].

The user interface to the low-fidelity microsimulator will provide a means for specifying computational nodes (CPNs), load balancing parameters, diagnostic outputs, and simulation outputs, as well as data sources to be used as inputs.

Figure 6 shows how the microsimulator might use other TRANSIMS components.

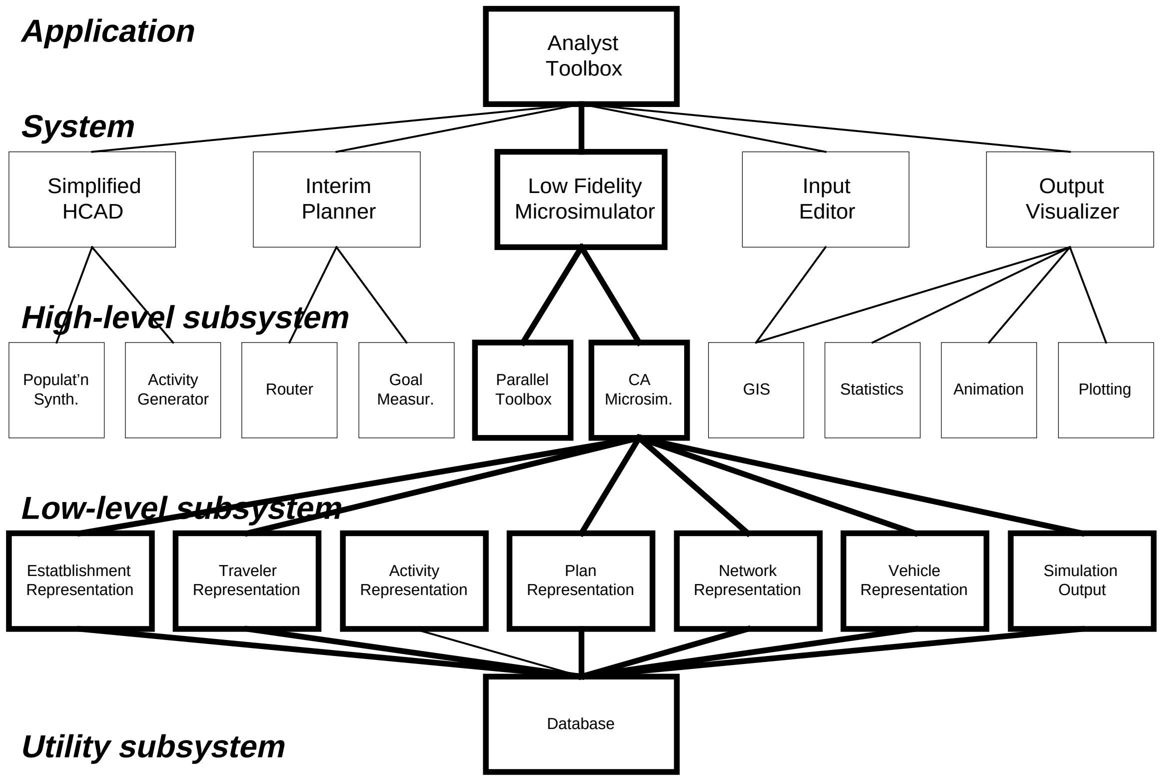

Figure 6. Software components used by the low fidelity microsimulator.

5. Output Visualizer

The output visualization system provides an integrated interface to the various tools available for viewing and analyzing aspects of simulation and plan data. There are five types of output available:

- animation
- statistical analysis
- geographic analysis
- plots
- data export

The user can select a geographic region of interest (and time interval) and filter the available data before visualizing it. The output visualization system will retrieve the specified data from the data sources and then send the data to the appropriate visualization tool. The output visualization system interface will handle as much of the user interaction as possible. Some of the interaction may have to take place in the third-party software that is integrated into the output visualizer.

The results of statistical and geographic analyses can also be plotted. It must also be possible to export data to standards formats for spreadsheets, etc.

C. High-Level Subsystem Layer

The high-level subsystems each provide services to one or more TRANSIMS systems. These subsystems enhance the reusability and flexibility of the software. For IOC-1, there are the following high-level subsystems:

- Population Synthesizer
- Activity Generator
- OD Matrix/Route Disaggregator
- Router
- Parallel Toolbox
- CA Microsimulation
- GIS
- Statistics
- Animation
- Plotting

1. Population Synthesizer

Using 1990 census, data a baseline synthetic populations of households will be generated which statistically mimic those sampled in the 1990 census. These populations will be produced on a census tract or block group basis. Each household and person in the synthetic population will be associated with the entire suite of socioeconomic characteristics available from the census. For future applications, these households will be aged to the desired date using projected land use and demographic trends in the study region. The populations will not be aged for IOC-1.

Input data for the generation of the baseline population will include the Census Bureau Standard Tape File 3 and the Public Use Microdata Sample. Standard statistical techniques will be used to generate the synthetic populations.

For IOC-1 baseline synthetic populations will be generated off-line. Multiple populations will be produced, but the software for doing so is not part of IOC-1. The households of these populations will be placed at random at locations in the census tract or block group.

2. Activity Generator

Travel activity will be predicted for each household and household member using national trends and local activity surveys. These desired travel activities will be passed to the planner for routing and scheduling. Activities will be assigned to either the household or the individual. For example, work activities would be assigned to the individual, while a household activity could be a shopping trip. Additionally, activities will be either mandatory or discretionary.

The structure of IOC-1 will allow for future activity-based travel. However, for the applications considered in IOC-1, travel activity will be replaced by trips that are generated from OD matrices. All trips will be assigned to individuals (no household activity analysis is planned for IOC-1) and all trips will be treated as mandatory. Trips from an OD matrix will be matched with the demographics for the households. These trips will be randomly distributed to individuals from demographically matched households within the census tracts or block groups that make up the Dallas Travel Survey Zones.

3. OD Matrix/Route Disaggregator

For IOC-1, actual travel data is available only in the form of origin-destination (OD) matrices and, possibly, traffic and turn counts. To produce trips, a utility preprocessor will be needed to disaggregate OD matrix zone-based traffic flow down to individual travelers, specific routes and activity addresses. Methods have currently been examined that will disaggregate the zonal flows along the boundaries of the detailed area of interest. Once the individual travelers are placed on routes that enter the area of interest and assigned starting times, the simplified planner will then perform the normal route assignment to generate detailed individual trip plans. The travelers will move to final destinations within the area of interest or through and out of the area of interest based on their aggregate OD matrix assignments. If traffic or turn counts are available for the area of interest, they will be used to calibrate the planner's preference distributions.

4. Router

Given household or individual activity demand, individual travel behavior, and individual travelers along the boundaries of the detailed area of interest, this subsystem will provide feasible or optimally infeasible sets of trip plans. A feasible trip plan is one in which all individual goals have been satisfied; an optimally infeasible plan is one that has not satisfied all of the individual's goals but that has minimized the non-zero goal deviations.

The router will be structurally consistent with the mode, route and activity assignment enhancements planned for later IOCs.

5. Parallel Toolbox

The parallel toolbox subsystem is those parts of the low-fidelity microsimulator that deal with running on a parallel distributed computer. The parallel toolbox provides a master/slave parallel computing model implemented on top of the PVM toolkit for heterogeneous network computing. PVM provides the message-passing substrate that allows tasks on different machines (CPNs) to communicate. Using the parallel toolbox, dynamic load balancing of the road network and associated vehicle objects is provided and is based on usage statistics gathered while the simulation runs. CPNs may be added or deleted during a running simulation. Diagnostic outputs will be provided, as will some form of fault tolerance. Simulation outputs will be produced in parallel.

6. CA Microsimulation

The CA (cellular automaton) microsimulation subsystem is those parts of the low-fidelity microsimulator that deal with doing traffic simulation using cellular automata. The network representation in this approach is grid based, and a mapping from the general network representation subsystem to grids will be provided. Similarly, mappings from the vehicle and traveler representation subsystems into vehicle/driver combinations that are suitable for the CA approach to vehicle motion will be made. Travelers will utilize plans from the plan representation subsystem.

7. GIS

The geographic information subsystem (GIS) provides high-level support for the following functions:

- editing network data
- viewing all types of geographic data and non-geographic data that can be linked to geographic data
- thematic display and analysis of plans and simulation output
- aggregation/disaggregation of geographic data
- preprocessing/formatting of geographic data from external sources (i.e., MPO data)

The GIS subsystem will support the import/export and editing of data available from other data-related subsystems (database, network, traveler, vehicle, plan, and output) as well as supporting the export of data to visualization and analysis tools such as the plotting, statistics, and animation subsystems.

8. Statistical Analysis

The statistical analysis subsystem will support the following general types of statistical analyses:

- confidence intervals
- analysis of variance
- hypothesis testing

All of the data available from other data-related subsystems (database, network, traveler, vehicle, plan, output, and GIS) will be importable for analyses. Predefined and user-definable analyses will be available for computing various measures of effectiveness; it will also be possible to save an analysis configuration for later recall and re-running. The results of analyses will be exportable to the plotting subsystem.

9. Animation

This subsystem will provide animated display of vehicle movement and traffic controls (e.g., signals) in real time and in accelerated time. Other map features such as buildings, bodies of water, topography will not be displayable for IOC-1. The vehicles and network features will be selectable with the mouse to obtain detailed attribute data. It will be possible to color-code vehicles based on their attributes. Plans can also be animated.

10. Plotting

The plotting subsystem will support the following general types of plots:

- scatter plot
- histogram
- lines
- areas

Three-dimensional and color plotting will be supported, as will be the grouping of attributes. All of the data available from other data-related subsystems (database, network, traveler, vehicle, plan, output, GIS, and statistics) will be importable for plotting. Predefined and user-definable plots (whose configuration can be saved and loaded) will be available. It will also be possible to customize axes, legends, markers, and annotation.

The specific types of plots will include the following:

- waterfall plots
- fundamental diagrams (all styles)

D. Low-Level Subsystems Layer

The low-level subsystems provide basic services (mostly data and operations on data) to the high-level subsystems and to the systems. The representation[1] subsystems model the basic attributes and behavior of transportation objects for the high-level subsystems and for the systems; they also supply a common representation of objects that are used throughout TRANSIMS. These subsystems never interact directly with the user. For IOC-1, there are the following low-level subsystems:

- Establishment Representation
- Traveler Representation
- Plan Representation
- Network Representation
- Vehicle Representation
- Simulation Output

The representations have been designed for long-term usefulness in TRANSIMS, not just for IOC-1.

Different systems and high-level software components need different views of the basic objects: The basic representation of an object contains the attributes and behavior that all subsystems using the object need. The various views of the object in different subsystems also contain the object's additional attributes and behavior specifically required by that subsystem. For example, the CA microsimulation subsystem does not need to know the demographic characteristics of a traveler, but the Statistics subsystem probably does. It would be inefficient to carry all of the demographic information in a microsimulation, however.

---

[1] A *representation* is a view of a database, plus utility classes for using the domain data. The specification of a representation says nothing about the actual data organization. Rather, it specifies what data must be available. Functions for both reading and writing data must be a part of a representation. The representation subsystems will also contain utility functions for manipulating and using the data.

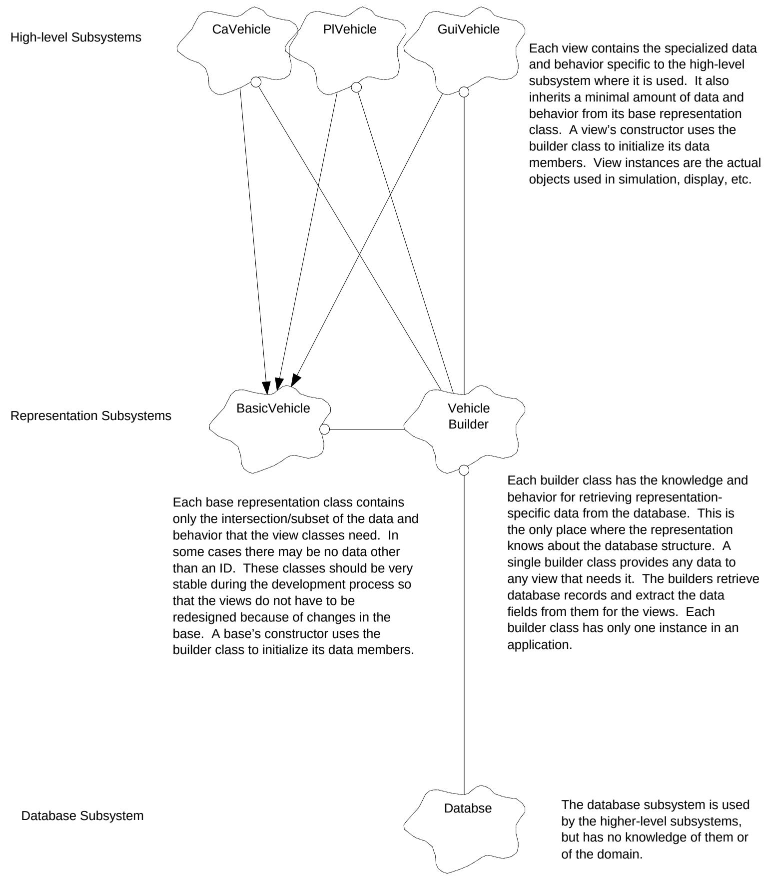

Figure 7. Example of a basic representation of a class and a view of it (Booch notation).

1. Network Representation

The transportation network representation includes detailed information about roads, intersections, signals, sensors, transit systems, rail infrastructure, bikeways and walkways. Network topology is represented along with attributes that describe the nodes and links in the network. Multiple views of the network representation are required, in some degree, by the planner, microsimulator, and output visualizer.

Link attributes for the road network include such characteristics as link type, length, directionality, speed limit, number of lanes, special lane designators, grade, toll, passing allowed, visibility range, street name, traffic capacity, etc. Node attributes may include node type, associated intersection (if applicable), etc. Provision will be made to represent the vehicular traffic network at a resolution at least as detailed as that of the standard traffic engineering tools TRAF and TRANSYT-7F.

Intersections may be represented at multiple levels of fidelity. The planner requires less fidelity than the low-fidelity microsimulator, which uses a medium-fidelity representation incorporating multiple queues. This intersection representation will include algorithms for handling the queues. Signalized and unsignalized intersections will be represented, and the representation must include lane usage and allowable turning movements. Both timed and actuated signals will be included, along with sensors for actuated signals. The signal representation will include algorithms for cycling through signal phases.

Transit networks, associated transit schedules, and intermodal transfer facilities will not be represented in IOC-1, but provision will be made to include them in a later IOC.

The user interface to the network representation will be provided through the Input Editor. The network subsystem will support the importation of external data in formats such as Arc/Info, TRANSYT-7F, and the TRAF family.

2. Establishment Representation

The definition of an establishment includes households, group quarters, and businesses. Each establishment will posses a unique identifier and the socioeconomic attributes required by the activity generator subsystem.

IOC-1 will not focus on commercial activity or the movement of freight. Provisions will be made for the needs of the more sophisticated planner and disaggregator to be developed in a later IOC.

3. Traveler Representation

The traveler representation includes the demographic, socioeconomic, and geographic attributes needed for identifying travelers and for planning trips. It also includes the driver representation and driver model needed for simulating driver behavior. The external data resources will not be available for the simplified planner. Thus, the traveler representation for IOC-1 will be fairly simple. Nevertheless, it should implicitly specify unique travelers and make it possible to associate them with vehicle identifiers in the plan representation. The framework developed will be consistent with the future enhancements to be made in the planner and disaggregator.

The driver representation describes the attributes of the drivers used in the models. Driver's decision-making processes may not be modeled at the same level of fidelity in every simulation, so a flexible driver representation that supports a particular decision logic without requiring unnecessary attributes is needed. Driver attributes will be specific parameters that are required by the decision logic algorithms rather than abstract behavioral attributes such as aggressiveness.

Potential attributes include driver age, sex, socioeconomic attributes, desired speed, following distance, and acceptable gaps for left and right turns, for crossing intersections, and for changing lanes. Algorithms that define the driver's decision logic are also part of the driver representation. Traveler attributes required by the planner include household id, mandatory activities, age, sex, socioeconomic class, trip goal weights, and mode/route preference distributions. Driver attributes included in the current CA microsimulation include desired speed and a variable that causes drivers to vary from their desired speed some of the time.

The user interface to the driver representation will support user specification of values for the attributes required in the study as well as the ability to read existing descriptions of driver properties.

4. Vehicle Representation

The vehicle representation describes the attributes of the vehicles used in the models. Not all potential attributes are needed for every type of study (e.g. air quality studies require dynamic information about engine properties, while other studies may not). The vehicle representation should be flexible enough to support a variety of studies but not require unnecessary attributes for a particular study.

Potential attributes include such properties as vehicle type, maximum speed, maximum acceleration, maximum deceleration, stopping distance, length, width, weight, age, fuel type, and engine properties. Dynamic attributes such as position along road segment, lane, velocity, acceleration, and engine temperature may also be required. Algorithms that define the motion of vehicles are also part of the vehicle representation.

The version of the planner to be used in IOC-1 requires only vehicle type. The current CA microsimulation requires only maximum speed, but length will also be required if multiple vehicle types are supported. Acceleration and deceleration parameters may also be desirable to smooth speed fluctuations.

The user interface to the vehicle representation is provided through the input editor and will support user specification of values for the attributes required in the study, as well as the ability to read existing descriptions of vehicle properties. When multiple types of vehicles are modeled, fleet mix will also be under user control.

5. Plan Representation

The plan representation will provide a view of trips and trip chains in the form of routes, origins, and destinations during specific time periods. A trip chain is a sequence of trips. A trip must provide:

- trip purpose
- unique identifier for the traveler
- unique identifier for a vehicle
- starting node in the network
- desired departure time from that starting node
- destination node in the network
- desired arrival time at that destination node

Detailed routes will specify which streets simulated vehicles will follow in the microsimulation and when they should be at various points along the way to satisfy an associated trip goal.

The detailed route design will impact the complexity of both the microsimulation and the visualization systems. The actual format might not include the ID of every link and node along the way. Perhaps a detailed route should be included as part of the trip specification.

6. Simulation Output

The simulation output subsystem will gather the data generated by simulations and provide access to it for other subsystems as soon as the data is received. The simulation output will be configurable, and the following predefined configurations will be provided:

- trajectory information (time, segment, lane, position along segment, velocity, and acceleration)
- control systems (signals, sensors, and high-occupancy vehicle lanes)
- vehicle state information (brakes on, lights, and signaling)
- measures of effectiveness (vehicle-hours traveled, vehicle-miles traveled, average speed, average density, and headway)
- animation output (trajectory and control systems)
- engine performance (idle time, start time, stop time, temperature, and fuel consumption)
- emissions (CO, NO_x, O_3, particulates, and aerosols)
- traveler characteristics (vehicle occupancy, demographics, trip purpose, and plan fulfillment)
- additional outputs available from TRAF and TRANSYT-7F products

This subsystem will utilize the database subsystem to support metadata, data distribution, data export, and archiving. Special provision will be made for dealing with the large amount of data generated by simulations. It will also be possible to perform compression (lossy or lossless) on the data to reduce the storage required for it. (Only limited capabilities will be developed in this area for IOC-1.)

In IOC-1, no automated feedback mechanism will be provided for using simulation output to alter simulation or planner input for subsequent simulations. This can be accomplished, manually, through a (possibly complicated) series of ad-hoc queries involving the output and input data sets.

E. Utility Subsystem Layer

The utility subsystems provide basic domain-independent services to the higher level components of TRANSIMS. These are used to isolate the rest of TRANSIMS from dependence on the specifics of the operating system, file system, communications network, etc. The components of this layer include:

- Database
- Interprocess Communication
- Containers

1. Database

The database subsystem provides low-level services for accessing and modifying data. It forms a layer separating the other subsystems from the actual data files—the other subsystems will not have access to the data files at the physical level.

Each data source will be indexed by a unique (primary) index. Additional (secondary) indexes will be allowed. The interaction between other subsystems and the data will be mediated by the public interfaces of the classes in the database subsystem. Range lookup of key values and ad-hoc SQL queries will be supported. Procedures (e.g., C++ templates, preprocessor macros, or custom preprocessor) will be available to coordinate the maintenance of the database schema and class definitions. The database subsystem will maintain metadata specifying the following information for all of the data sources:

- existence
- versions
- network location
- attributes (data dictionary)

The distribution of a data source over the network will not be supported for IOC-1, although this capability will be added in a later IOC. Migration of data will be supported, but the database subsystem will not be required to automatically load-balance the data among the network nodes. An archiving facility will also be available. There will also be a mechanism for extracting data specified at run-time from a data source and exporting it to a formatted binary or ascii file.

2. Interprocess Communication

A C or C++ interprocess communication library will provide support for the communication between distributed TRANSIMS components running on different machines or in different address spaces.

3. Containers

A C++ container class library will provide support for grouping object in sets, bags, lists, graphs, etc.

III. Implementation

A. Development Environment

1. Supported Platforms

The initial development of TRANSIMS will take place on a Sun hardware/software platform. The development work will be portable to other Unix platforms, but not necessarily immediately, insofar as compiler technology currently varies widely from platform to platform. Windows NT will be supported for the user interface, but not for the planner and microsimulation; Windows, DOS, and Macintosh will not be directly supported, although they may be used as X clients for the user interface.

2. Software

The SunPro C++ suite of C++ development tools will be used for work on the Sun hardware/software platform. This suite includes a compiler, debugger, source code editor, and object-oriented browser. The Borland integrated C++ development environment will be used for Windows NT development. Both the SunPro and Borland compilers are certified for use with the C++ class libraries recommended below. The configuration management tool's version of "make" will be used for building executables. Purify will be used for debugging memory management. Software development team members, of course, have the option of using any compatible tools that supplement the above.

The word processor standard will be FrameMaker. The standard tool for object-oriented software engineering will be Rose/C++. The configuration management tool will be ClearCase.

B. Tools (Third-Party Software)

1. C++ Libraries

The ANSI standard C library [Pl 92] and the ANSI draft standard C++ library [Pl 95] will be used. POSIX-compliant operating system calls will also be used [Ga 95]. This combination of standard libraries will support the portability among Unix platforms, and between Unix and Windows NT.

The Booch Components [RW 94] will be used for container classes and exception handling. These are robust, easily usable, well-designed, and portable.

PVM will be used for distributed application development; it is portable among Unix platforms. RPC will be used to support interprocess communication; it is portable among Unix platforms, and between Unix and Windows NT.

The Rogue Wave DBtools.h++ library [SL 95] will be used for database access. It provides a C++ interface to databases such as Oracle and Sybase.

2. Graphical User Interface

ArcView will be the primary graphical user interface; it will be customized with Avenue to provide a uniform interface to as many of the TRANSIMS subsystems as possible. It will serve as a launcher, viewer, and data manipulator. It may be necessary to use ArcInfo for the parts of the graphical user interface where the customization capabilities of Avenue are insufficient.

3. Database

IOC-1 will use the Oracle 7 relational database to manage the majority of the data in the database subsystem. This interfaces seamlessly with ArcView and ArcInfo. A customized storage system will be required to compress large simulation output data sets to manageable size.

4. Statistics

The ArcInfo-compatible version of S+ will be used for statistical analysis.

5. Geographic Information System

The Unix version of ArcInfo 7 will be used for the processing and analysis of geographic data; one floating license will be required. The majority of the GIS functionality will be in ArcView, with ArcInfo used as a calculation engine.

C. Overview of Application Programs

Figure 8 provides an overview of the application programs used for IOC-1—both commercial products and those developed by LANL. Figure 9 shows how these programs relate to the TRANSIMS analyst toolbox and the top level of the TRANSIMS architecture.

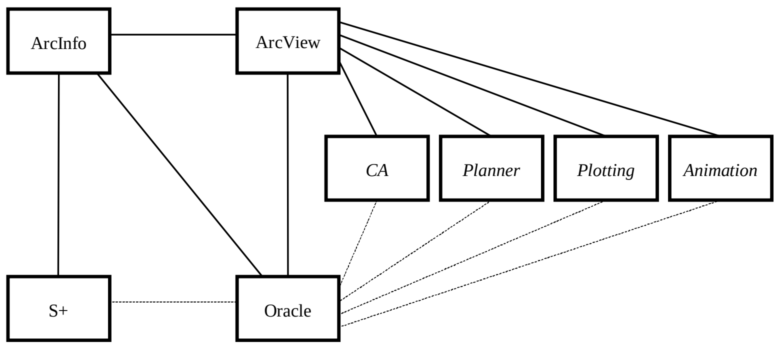

Figure 8. Overview of application programs in TRANSIMS. Those in italic type are LANL-developed products, and those in upright type are commercial products customized by LANL. The solid lines represent strong links, and the dashed lines represent weak links.

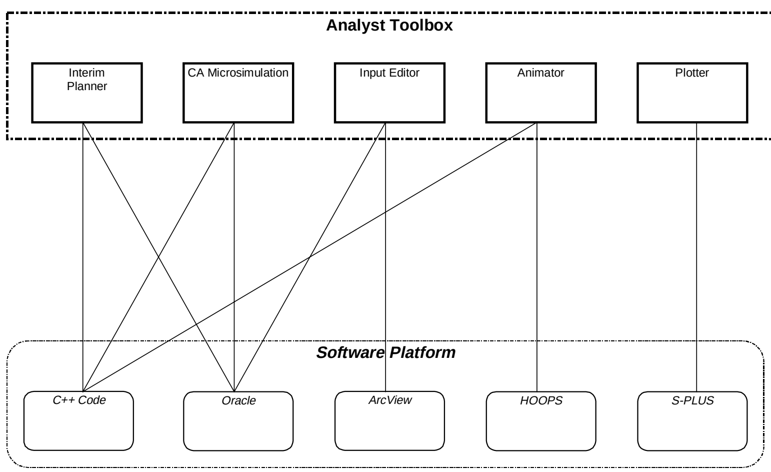

Figure 9. Relationship between application programs and the TRANSIMS architecture.

D. Standards

1. Coding

We recommend adopting the coding (*naming* and *usage*) standards presented in *Taligent's Guide to Designing Programs* [Ta 94]. Supplemental usage guidelines are available in the books by *C++ FAQs* [CL 95], *C++ Strategies and Tactics* [Mu 93], *C++ Programming Style* [Ca 92], etc.

Additional coding recommendations are:

- Classes have names indicating in which subsystem they belong. This will aid in navigating source code.
- Formal parameters for member functions should be included in the class definitions. This makes the class definition more understandable.
- Member functions should not be inlined in the class definition. This makes the definition less cluttered and makes it easier to switch from inlining to not inlining.
- The documentation for each class should include a general description of the services it provides and should specify any invariants or states it has.
- The documentation for each member function should include a general description of its function, define its arguments, specify its pre- and post-conditions, say what state changes may occur, and specify what exceptions it throws and why.

2. Documentation

The TRANSIMS documentation will describe TRANSIMS at a level of detail that is useful to the software developer and, at the same time, understandable by the non-programmer. It will consist primarily of English text, supported and augmented by the diagrams discussed below. Refer to *Using the Booch Method* [Wi 94] for more detail.

The design documentation will include the following:

- The *architectural specification* captures the major abstractions and design decisions that apply to the whole system.
- *Class-category diagrams* partition the system into high-level groupings of classes and show their visibility to each other.
- *Design class diagrams* identify the key classes of the domain and show the relationships between classes. They also show the abstractions of the physical implementation, detailed data types and structures, and the mapping of the logical abstractions to the physical abstractions.
- *Design class specifications* define the class; its relationships, attributes, and superclasses; and the interface of its operations. They also show domain and implementation details, such as algorithms for complex operations, internal data attributes, and access control for operations and data.
- *Design object-scenario diagrams* illustrate how the objects will interact when tracing the execution of a use case. They also show the full implementation detail of a key mechanism, including objects that deal with I/O, data structures, and persistent data.
- The implementation documentation will consist of the following:

- *Documented source code.* Source code is documented according to standards defined above.
- *Test results.* Results from carrying out the test suite are saved for regression testing in subsequent executable releases.
- *User manuals.* User documentation provides sufficient detail for the non-programmer to successfully exercise the software. The documentation will be available on-line.

3. Configuration Management

All source code, data files, diagrams, and documents should be under configuration management control where practical. One team member should be the primary contact for dealing with configuration management issues. A configuration management plan should be completed as soon as possible.

E. Process

1. Methodology

We will use the Booch methodology as presented in John Berry's class notes [CS 94], Iseult White's primer [Wi 94], and Grady Booch's text [Bo 94]. To reduce risk, we emphasize the following points from *Managing Development*, Chapter 8 of John Berry's class notes:

- We must fully utilize iterative development; we should have an incrementally improved executable release every three to six weeks.
- We must preserve a clean, layered internal system structure (architecture).
- The individuals responsible for the TRANSIMS architecture must continue to maintain the architectural integrity of the system. Specifically, the team should approve all changes to architectural interfaces, help assess project risks, and help schedule the order and the content of the iterations.

To coordinate the development effort, we will use Iseult White's book as a primer, especially for identifying project deliverables. Although we will produce simplified versions of some of the documents, we should follow all of the steps of each iterative cycle and produce a new executable version every few weeks. For efficiency, we should use the Rose tool as fully as possible and stick to its version of Booch notation and C++ coding style. These match White's book. We should rely on the professionalism of the team members rather than spend time overspecifying the methodology.

2. Reviews

We recommend the adoption of a software inspection procedure along the lines of that outlined by Steve McConnell in §24.2 of *Code Complete* [Mc 93]. Inspections have been shown "to be extremely effective in detecting defects and to be relatively economical compared to testing" [Mc 93]. Each inspection involves a moderator/scribe, author, and 1–4 reviewers. Each reviewer works alone inspecting the code for 90 minutes. Later the participants meet and the author paraphrases the code design and logic. Errors and their severity are noted by the reviewers, but they are not discussed beyond identifying them.

The length of the meeting is limited to two hours, and the moderator writes an inspection report. Informal discussion of solutions is postponed until after the meeting.

Reviews will be held for both the design and the implementation phases of development. Reviewers external to TRANSIMS may be included on the review panels.

3. Testing

It is recommended that each unit (class or group of classes) be tested by the author or other interested parties before the code is checked into the configuration management system for use by others. (All test documentation, code, and data will be under configuration management control.) All functions and branches within functions will be tested.

Integration testing will take place as soon as sufficient units are available to integrate the components. System testing will take place once per iteration of the software development cycle.

IV. Future Work

The TRANSIMS software architecture will be expanded, revised, and updated throughout the lifetime of the TRANSIMS project. For example, in IOC-2, support will be added for regional and intermodal traffic simulation.

V. References

[Bo 94] G. Booch, *Object-Oriented Analysis and Design with Applications*, (Redwood City, California: Benjamin/Cummings Publishing Company, 1994).

[Ca 92] T. Cargill, *C++ Programming Style*, (Reading, Massachusetts: Addison-Wesley, 1992).

[CL 95] M. P. Cline and G. A. Lomow, *C++ FAQs*, (Reading, Massachusetts: Addison-Wesley, 1995).

[CS 94] Catalyst Solutions, *Object-Oriented Analysis and Design with C++*, (n.c.: n.p., 1994).

[Ga 95] B. O. Gallmeister, *POSIX.4: Programming for the Real World*, (Sebastopol, California: O'Reilly & Associates, 1995).

[Mc 93] S. McConnell, *Code Complete*, (Redmond, Washington: Microsoft Press, 1993).

[Mu 93] R. B. Murray, *C++Strategies and Tactics*, (Reading, Massachusetts: Addison-Wesley, 1993).

[Pl 92] P. J. Plauger, *The Standard C Library*, (Englewood Cliffs, New Jersey: Prentice, 1992).

[Pl 95] P. J. Plauger, *The Draft Standard C++ Library*, (Englewood Cliffs, New Jersey: Prentice Hall, 1995).

[Ri 94] M. Rickert, "A First Draft on how to Integrate High Fidelity and Cellular Automata Approaches to Microsimulation in TRANSIMS on a Distributed Computer Network," Version 1.0, July 18, 1994.

[RW 94] Rogue Wave Software, *The C++ Booch Components,* Version 2.3, (Corvallis, Oregon: Rogue Wave Software, 1994).

[SL 95] S. Sulsky and K. L. Lohn, *DBtools.h++ User's Guide and Tutorial*, Version 1, (Corvallis, Oregon: Rogue Wave Software, 1995).

[Su 95] S. Sulsky, *DBtools.h++ Class Reference*, Version 1, (Corvallis, Oregon: Rogue Wave Software, 1995).

[Ta 94] Taligent Press, *Taligent's Guide to Designing Programs*, (Reading, Massachusetts: Addison-Wesley, 1994).

[Wi 94] I. White, *Using the Booch Method,* (Readwood City, California: Benjamin/Cummings, 1994).